SCC-203 ALGORITMOS E ESTRUTURAS DE DADOS II
Prova 1 - Gabarito

Nome: _____ Nro. USP_____

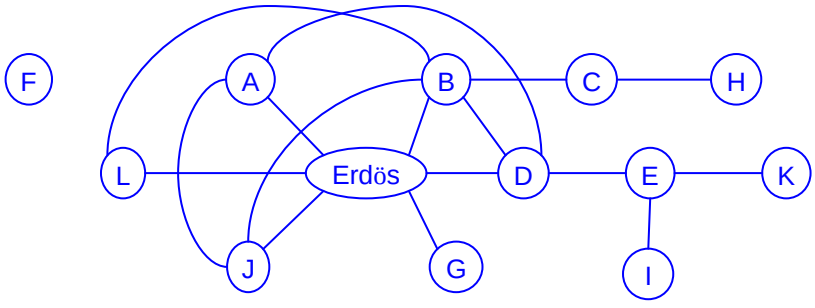
1) O matemático húngaro Paul Erdös (1913-1996), um dos mais brilhantes do século XX, é considerado o mais prolífico da história. Erdös publicou mais de 1.500 artigos, em colaboração com cerca de outros 450 matemáticos. Em sua homenagem, os matemáticos criaram o "número de Erdös". Todo matemático que publicou um artigo com Erdös tem número de Erdös 1. Os que não possuem número 1, mas escreveram um artigo com alguém que possui número 1, possuem número 2, e assim por diante. Quando nenhuma ligação pode ser estabelecida entre Erdös e um matemático, diz-se que este possui número de Erdös infinito. Por exemplo, o número de Erdos de Albert Einstein é 2; e, talvez surpreendentemente, o número de Erdös de Bill Gates é 4.
Considere um subconjunto de 12 matemáticos distintos identificados por A, B, ..., K e L. A lista abaixo informa os que têm artigos em comum:

- o autor A tem artigos com D e J;
- o autor B tem artigos com C, D, J e L;
- o autor C tem artigos com H;
- o autor D também tem artigos com E;
- o autor E também tem artigos com I e K.

Erdös tem artigos com os autores A, B, D, G, J e L (e, logicamente, cada um desses autores tem artigo com Erdös; idem para os demais casos).
Considere o problema de determinar o número de Erdös de um autor. Faça:

(a) (1.0) Modele a situação dos autores acima por meio de um grafo, indicando o que representam os vértices e as arestas (desenhe o grafo completo).

Possível solução: o problema representa um grafo não direcionado, com autores como vértices e arestas entre autores que publicaram juntos. O grafo pode ser ponderado ou não. Se for ponderado, o número de Erdös pode ser o peso associado às arestas. Opcionalmente, o número de Erdös pode ser associado a cada vértice.



(b) (1.0) Explique qual conceito/algoritmo de teoria dos grafos pode ser usado e como pode ser usado para resolver o problema.

O problema de descobrir o número de Erdös de um dado autor é equivalente a determinar o comprimento do caminho mínimo entre 2 vértices no grafo (no caso, os 2 vértices representam 2 autores, Erdös e o outro). Nesse caso, cada aresta percorrida em direção a um autor a partir de Erdös conta um a mais no número de Erdös do autor procurado. Qualquer algoritmo de caminho mínimo pode ser usado para resolver o problema, inclusive uma busca em largura a partir do nó Erdös.

(c) (0.5) Com base na especificação da questão, no seu modelo no item (a) e em sua argumentação no item (b), responda: qual o número de Erdös dos matemáticos C e F? Justifique sua resposta.

Analisando o grafo, observa-se que o autor C tem número de Erdös igua a 2, pois há um caminho mínimo de 2 arestas entre C e Erdös. Como F é um componente desconectado do grafo e não há um caminho mínimo entre Erdös e F, diz-se que F tem número de Erdös infinito.

2) O Algoritmo abaixo faz percurso em profundidade, pelos vértices de um grafo não direcionado dado, a partir de um vértice v.

```
Dado G(V,A):
Prof(v)
        marcar v;
        para cada w ∈ ListaAdjacencia(v) faça
           se w é não marcado então
                Prof (w);
fim /*Prof(v)*/
```

Como o algoritmo acima poderia ser usado/modificado para:

(a) (0.5) Verificar se o grafo G é cíclico ou acíclico. Justifique.

Resp.:
No trecho:

```
se w é não marcado então
                    Prof (w);
senão "o gráfico é cíclico"
```

Se um vértice for alcançado mais de uma vez no trajeto, então ele pertence a um ciclo.

(b) (0.5) Verificar se o grafo G é conexo ou não.

Resp.:

Ao chamar Prof(v) para um vértice v inicial, se, no final, ele não tiver passado por todos os vértices do grafo, então ele não é conexo. Ao chamar para os vértices restantes, até que todos tenham sido visitados, obtêm-se as componentes conexas do grafo

(c) (1.5) Considere agora a representação de grafos por listas de adjacências (declaração em linguagem C a seguir):

```
#define MaxNumVertices 1000

typedef int elem;

typedef struct no_lst {
    int v;
    elem peso;
    struct no_lst *prox;
} no_lista;

typedef struct {
    no_lista *LAdj[MaxNumVertices + 1];
    int NumVertices;
} Grafo;
```

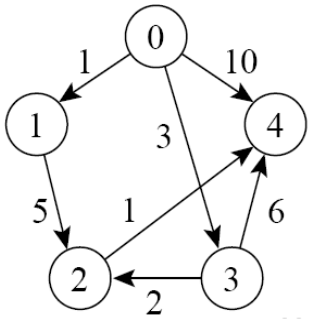
Implemente em C uma função que retorna o peso de uma aresta (V1,V2) no grafo direcionado, caso ela exista. Assumimos que os pesos são valores >= 0. Assim, a função deve retornar 0 se a aresta não existir, retornar o seu peso se ela existir, ou retornar -1 se ocorrer algum erro. A função **deve ter** o seguinte cabeçalho:

```
int peso_aresta(Grafo *G, int V1, int V2)
```

Possível resolução:

```
int peso_aresta(Grafo *G, int V1, int V2) {
    if ((V1 < 1) || (V1 > G->NumVertices) ||
        (V2 < 1) || (V2 > G->NumVertices)) {
        return -1;
    } else {
        int encontrou = 0;
        no_lista *aux = G->LAdj[V1];
        while ((aux != NULL) && (!encontrou))
            if (aux->v == V2)
               return (aux->peso);
            else
                aux = aux->prox;
        return (0);
    }
}
```

3) Considere o dígrafo valorado abaixo:



Faça:

(a) (1.0) Aplique o algoritmo de ordenação topológica no grafo. Considere as listas de adjacência ordenadas crescentemente pelos números dos vértices, e use uma pilha para armazenar os vértices com grau de entrada igual a zero.

Ordenação Topológica Resultante: 0, 3, 1, 2, 4_____

(c) Considerando a implementação de dígrafo com matriz de adjacência, faça:

i) (0.5) Mostre a Matriz Adjacência, X, do dígrafo

0 1 0 3 10
0 0 5 0 0
0 0 0 0 1
0 0 2 0 6
0 0 0 0 0

ii) (0.5) Mostre a Matriz Caminho, P, do dígrafo, conforme é calculada pelo Algoritmo de Roy-Warshall (RW).

0 1 1 1 1
0 0 1 0 1
0 0 0 0 1
0 0 1 0 1
0 0 0 0 0

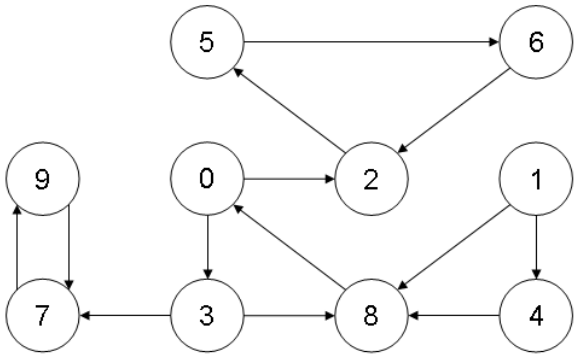
iii) (1.0) Mostre a Matriz dos Custos dos Caminhos Mínimos, MC, do dígrafo, conforme é calculada pelo Algoritmo de Floyd-Warshall (FW).

∞ 1 5 3 6
∞ ∞ 5 ∞ 6
∞ ∞ ∞ ∞ 1
∞ ∞ 2 ∞ 3
∞ ∞ ∞ ∞ ∞

iv) (1.0) Mostre a Matriz de Especificação dos Caminhos Mínimos, M, do dígrafo, conforme é calculada pelo Algoritmo FW^{+}. Use -1 no lugar de 0, para não confundir com o vértice 0.

-1 1 3 3 3
-1 -1 2 -1 2
-1 -1 -1 -1 4
-1 -1 2 -1 2
-1 -1 -1 -1 -1

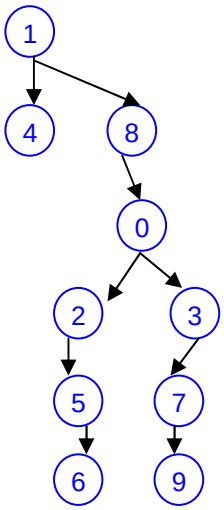
4) Considere o grafo abaixo. Considere também que as listas de adjacência estão ordenadas pelos rótulos dos vértices.



Faça/responda:

(a) (0.5) Monte a(s) árvore(s) de busca em largura quando a busca é feita a partir do vértice 1, e exiba a sequência dos vértices visitados.

Sequência: 1,4,8,0,2,3,5,7,6,9



(b) (0.5) Monte a(s) árvore(s) de busca em profundidade quando a busca é feita a partir do vértice 1, e exiba a sequência dos vértices visitados.

Sequência: 1,4,8,0,2,5,6,3,7,9